Biomedical Robotics course

University of Genoa

Mouse cursor control via body motion data

Report

Students: Aurora Bertino, Chiara Saporetti,
Gabriele Reverberi, Sara Romano

Year: 2020 - 2021

Contents

Abstract

This projects creates a simple mapping from the motion data of a smartphone's gyroscope to the motion of a computer's cursor. This has been implemented by building a Simulink model on Matlab and by connecting it to an Android smartphone via USB cable. Results were tested by positioning the device on the wrist and on the torso and they were visualized via a VR simulator block.
The whole project can be found at:
https://github.com/ChiaraSapo/Biomedical-robotics-Assignments/tree/main/Ass3

1 Introduction

1.1 Project's goal

The main goal of this project is to make a mapping from the motion data extracted from the gyroscope of a smartphone to the motion of a computer's cursor. This has been implemented by building a Simulink model and by connecting it to an Android smartphone via USB cable. The hardware and software tools will be analyzed in detail in the next section of this report.

In order to try the mapping, the most simple way is by grasping the smartphone with the hand and by moving it like a pointer. However, our main focuses are on either helping people with rehabilitation or providing computer access for people with mild to severe disabilities. For this reason, the motion data can be acquired not only by grasping the smartphone with the hand (which may be useful for rehabilitation), but also by positioning it on the torso, so that a person with reduced or no arm mobility can still move the cursor.

Why did we choose the torso? First of all, we chose it instead of the shoulder because we saw that it was difficult to correctly measure the velocity variations of the shoulder with the smartphone only. This was mainly because it's difficult to attach it in a proper way. But we still wanted to use a part of the body that allowed to use something different from the arm in order to encounter the need of amputees, for example. In fact, the reasons for an arm loss are various, and range from several problems with blood circulation that result in amputation, injuries from traffic accidents and military combat, cancer, birth defects etc. There already are different ways to help these people cope with ADLs in general, but autonomy also comes from using technology, and especially the computer, since it has become such an important part of our lives.

1.2 Hardware and software tools used

The hardware tools used for this project are:

- Computer ASUS VivoBook with MATLAB 2020b installed;
- Android smartphone Huawei P30: used to input motion data.
- USB cable to connect the device to the host computer.

To model, simulate and analyze the system, Simulink software has been used. Its interface, based on graphical block diagramming tool, allows to have a schematic way

to perform our project. To collect sensor motion data in the Simulink environment from an Android device, the add-on **"MATLAB Support Package for Android"** must be properly installed. The support package includes a library of Simulink blocks to use Android device sensors and network interfaces. Once the package has been installed, the smartphone must be configured too (by using *androidhwsetup*). [3] [2] The smartphone has been used as a Gyroscope sensor (see Figure 1), therefore the **Gyroscope block**, provided by the newly installed package, has been added in our model. Then, other features of Simulink have been used to filter the sensor data and monitor the signals.

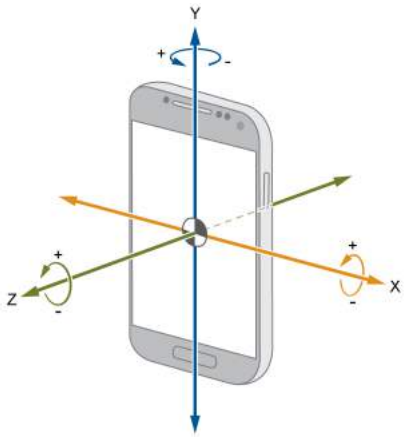

Figure 1: x, y, z angles in a smartphone

2 How to run our code

To run the project: download the MatlabCode folder, open Matlab and set the folder as its path, connect the smartphone to the computer via USB cable. Then type *Body_cursor_GUI* on Matlab's command window. Choose the desired part of the body you wish to try for our project and click it. The simulation will be run and after a few seconds it will be possible to move the cursor within the the visualization tool.

3 Description of the architecture

The complete architecture is given in Figure 2 and is useful to visualize the entire project that will be described deeply in the following subsections. The Simulink model of the Torso is the same as the model for the Wrist from a qualitative point of

view because the only difference is in the Gain's values of the Mapping module. This change guarantees an adequate Range of Motion for the selected body's part.
The model is divided into four subsystems:

- Data Acquisition
- Pre-Processing
- Mapping
- VR Simulator

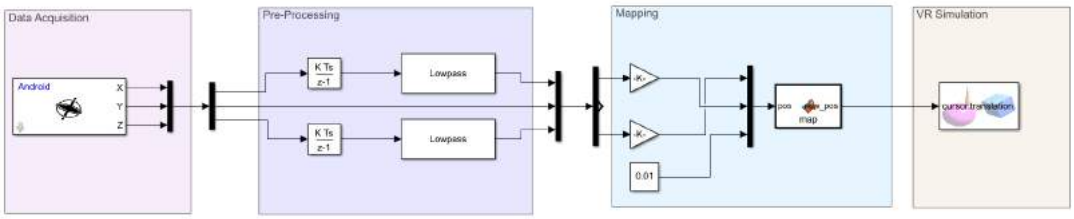

Figure 2: Simulink model of the architecture

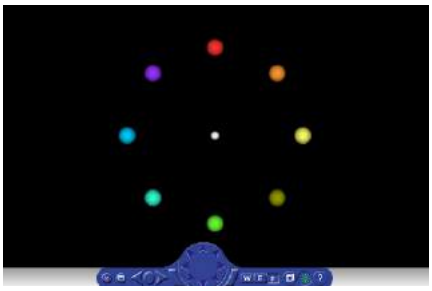

Figure 3: Eight balls simulation

The figure 3 shows the output of the VR simulator. The source file "8_palline_color.wrl" is given as parameter of the block and represents the virtual world associated to the model. Ultimately, the data from the gyroscope are used to control the cursor of this simulation. To have an input on the block, the cursor translation variables are enabled.

3.1 Smartphone signal output

The primary step to proceed with this project is to obtain a sufficient level of acknowledgement and familiarity with the data output from the Android sensors used. More specifically three Simulink blocks were taken into consideration: Gyroscope, Orientation and Accelerometer. For each block outputthe sampling time is set to 0,016 s, in order to have a sampling frequency of 60 Hz.

- Gyroscope: measures the rate of rotation around x, y and z axes, in rad/s. The output of the Simulink block is separated into 3 signals, one for each axis, each represented by a single precision scalar. To have a specific idea on the disturbance, a very simple Simulink block is implemented and shown in Figures 4 and 5.

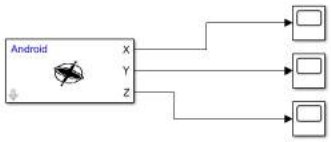

Figure 4: Simulink gyroscope test blocks

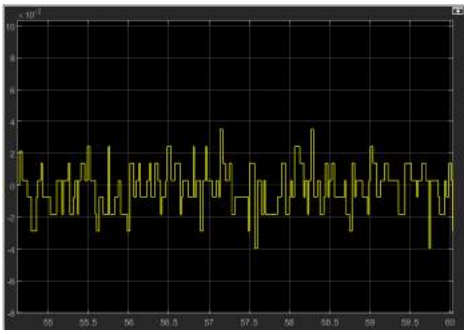

Figure 5: steady-device disturbance

As shown in figure 5 , the sampling time is 0.016 s and, by keeping the device perfectly steady, we can notice a disturbance oscillation with a maximum range of $\cong \pm 4 \times 10^{-3} rad/s$

- Orientation: measures the device rotation along the x(pitch) y(roll) and z (azimut) axes in degrees. The output of the Simulink block is an 1x3 array of single values. As for the case of the gyroscope, a simple Simulink model is implemented to roughy measure the signal disturbance.

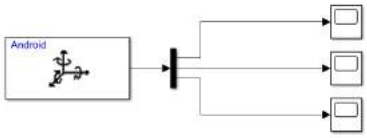

Figure 6: Simulink orientation test blocks

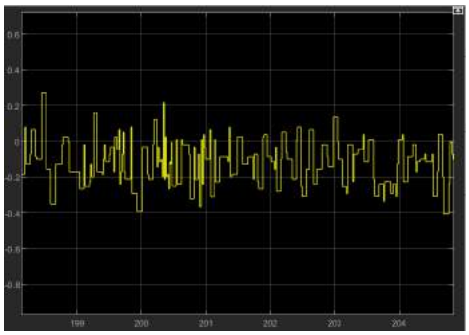

Figure 7: steady-device disturbance

As shown in figure 7, the sampling time is 0.1s and, by keeping the device perfectly steady, we can notice a disturbance oscillation with a maximum range of $\cong \pm 0, 4deg$

- Accelerometer : measures the linear acceleration along the x, y and z axes in m/s^2. Each x y z port outputs the acceleration as a single precision scalar . As before a simple Simulink model is implemented to roughy measure the signal disturbance

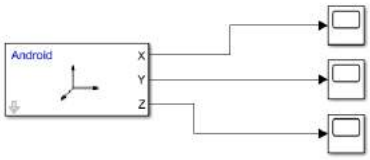

Figure 8: Simulink accelerometer test blocks

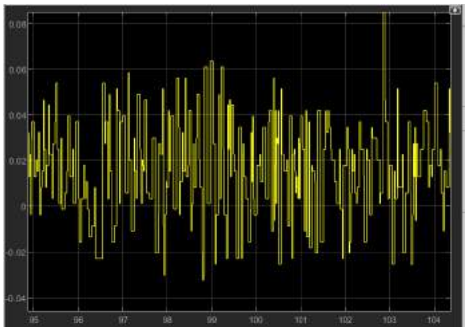

Figure 9: steady-device disturbance

As shown in figure 9, the sampling time is 0.016 s and, by keeping the device perfectly steady, we can notice a disturbance oscillation with a maximum range of $\cong \pm 0, 1m/s^2$.

Considering the disturbances described before and the fact that the animation we are willing to control has an order of magnitude of 0.1 (which refers to the distance between the center of the animation space and the coloured targets), data need to

be properly filtered, downscaled and integrated (the integration is not needed in the case of orientation data: the output are not accelerations nor velocities but angles).

One parameter for the choice between these three Simulink blocks was the **initialization process** needed:

- In the case of the accelerometer, data needed one further pre-processing step in order to separate the gravitational and body component, only taking the body component of the acceleration into consideration.
- In the case of orientation, the phone has to be put in a specific position every time, which does not always respond to the initial position of the animation.

To avoid these problems, the most suitable choice appeared to be the gyroscope module, that allowed to apply a simple, intuitive mapping from angles to positions without the need to insert an initialization part.

3.2 Sampling frequency

To avoid loosing data from the sensor input, we have to set a proper sampling rate. We did so by following the Nyquist criterion, for which the minimum value for the sampling rate is the Nyquist frequency, which is twice the highest frequency of the input signal - the *bandwidth*.

$$f_c > 2f_b \tag{1}$$

From an on-line literature review, we found out that the maximum sampling frequency for a smartphone's accelerometer is about 200Hz, so that was the highest value we could set. However, in this case we could obtain good results at lower frequencies and we chose a sampling frequency of 60 Hz, hence the simulation sampling time is set to 0.016 s.

3.3 FIR filter

The signal needs to be filtered since it is affected by errors due to thermal noise (fast vibrations). This is clear when we put the smartphone on the table: the output of the gyroscope is not zero. Visually, this means that the cursor of the VR simulator moves on the spot, which is an unwanted behaviour.
What we want is to have a good trade off between:

- allowing fluid movements of the phone in order to reach all targets;
- removing the movements of the white ball when the phone was stopped and in general "cleaning" the input signal;

And the solution was to use a low pass FIR filter to cut off undesired high frequencies.

In the ideal case a low-pass filter is only characterized by two different frequency responses : passband and stopband, and thus we did some trials with different values for the passband and stopband of the filter, to understand which frequencies should be removed. However, the filter also has other parameters to be taken into account. The parameters of a low-pass filter that describe its frequency response are the following :

- **Ripple**: deviation from flatness of the pase csband, it is a disturbance caused the real behaviour of the filter
- **Passband**: region in which the filter transfers the input as output (with an unitary gain)
- **Transition band**: frequency interval in which the filter is activated, at a shorter interval correspond a closer-to-ideal filter.
- **Stopband attenuation**: the value of the signal attenuation caused by the filter. It is equal to the difference between the lowest gain in the passband and the highest gain in the passband
- **Stopband**: frequency interval at which the filter should not transmit the input

To better understand the behaviour of the filter,the frequency response is the one shown in Figure 10.

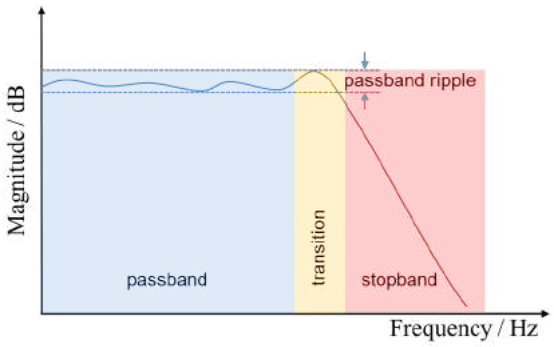

Figure 10: Frequency response of a generic FIR filter

3.3.1 Testing FIR filter parameters

The following tables describe some of the tests we performed in order to properly set the filter's parameters by directly connecting the filter's output to the VR simulator.

First trial:

Filter's parameters (1)			
Passband edge frequency	Stopband edge frequency	Maximum passband ripple	Minimum stopband attenuation
29 Hz	29,9 Hz	0,1 dB	60 dB

Result 1: Two trials have been done with these parameters:
In the first one, we moved the smartphone: the cursor on the VR Sink perfectly tracked the motions, but with a too high sensitivity.
In the second one, the smartphone was left on the table: here we could notice that the cursor started to move after a while. We can see this in Figure 11: The x and z gyroscope data are filtered but they are not zero (considering that the phone was not moving).
As we have previously written, this movement was due to the high frequency of the smartphone errors, which we want to avoid.

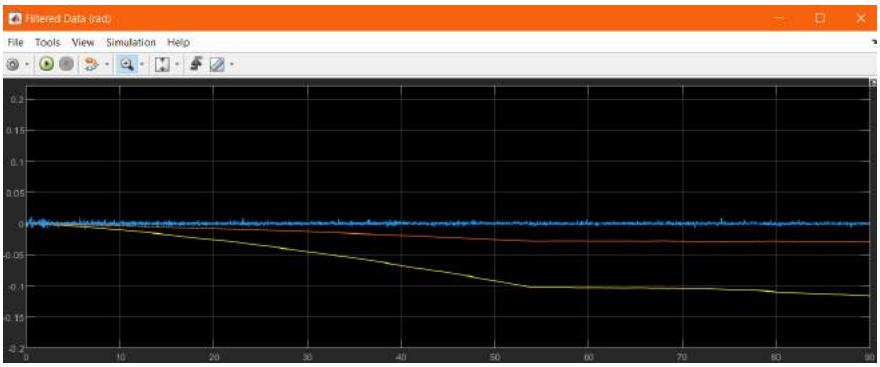

Figure 11: First Trial: Filtered Data scope

Trial 2:

Filter's parameters(2)			
Passband edge frequence	Stopband edge frequency	Maximum passband ripple	Minimum stopband attenuation
25 Hz	29 Hz	0,1 dB	60 dB

Result 2: We made two tests as in the first trial, and we encountered the same problem: since we attenuated more frequencies, the values of x and z went far from zero slower than the previous trial. However, this remains an unwanted scenario.

Trial 3:

Filter's parameters (3)			
Passband edge frequence	Stopband edge frequency	Maximum passband ripple	Minimum stopband attenuation
10 Hz	15 Hz	0,1 dB	60 dB

Result 3: body's motion is reproduced more accurately, but the cursor still moves a little bit over time.

Trial 4:

Filter's parameters (4)			
Passband edge frequence	Stopband edge frequency	Maximum passband ripple	Minimum stopband attenuation
7 Hz	10 Hz	0,1 dB	60 dB

Result 4: body's motion is reproduced more accurately compared to the previous test and is considered good. When the phone is in a steady state the cursor doesn't move. We carried out the test over a time of 300 s.

Note: By increasing the simulation time and waiting for some seconds, the error will always be there: we cannot avoid that, because the gyroscope always determines

the new position with respect to the previous one. As we previously wrote, to completely cancel this problem, we would need to use another sensor coupled with the first (i.e. an accelerometer), with a Kalman-filter approach.

For the **Passband Ripple**, the default Simulink value has been used. However, it could be updated in case this project was implemented with a real electronic filter and not with a simulated one.
The **passband** and **stopband** frequencies have been chosen between 7 and 10 Hz, a wide range purposely chosen not only to cut the sensor idle disturbancies (keeping the 60 Hz of the initial sensor oscillation in the cut-off part) , but also to gradually reduce body oscillations maintaining a proper hand-eye coordination and feedback.
The **stopband attenuation** value has been chosen to be 60 dB in order to guarantee a proper cut-off without increasing too much the disturbancies on the stopband (with a steeper filter, a higher disturbance frequency on this band is encountered).
As a result, the frequency response is the one shown in Figure 12.

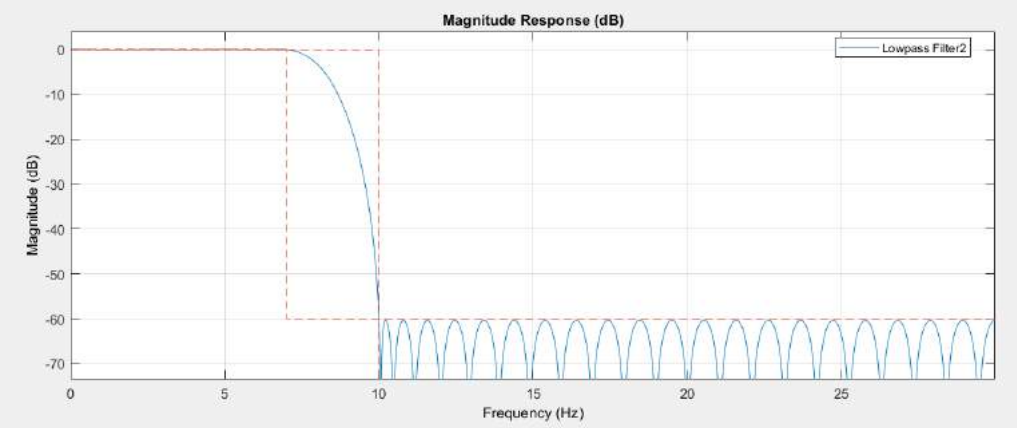

Figure 12: Frequency response of our FIR filter

3.4 Range of Motion

The sensitivity of the device is adjusted according to the type of motion that the person performs and to the body's part on which the device is placed.
Therefore, it has been modulated according to the movements that the person is able to perform.

- For the experiment executed on the wrist, we decided to set up a Range of Motion of **120 degrees**: [-60,+60] that allows a full motion of the wrist.

- For the experiment executed on the torso, we decided to decrease the Range of Motion to **40 degrees**: [-20,+20] since this body's part has less mobility.

The control scheme that allows this behavior is described by the following equations, in which the maximum angle (60 degrees for the wrist and 20 degrees for the torso, expressed in radians) is equivalent to the maximum displacement on the VR Sink axis (0.1 on both x and y axes since the 0.1 is the circumference radius on the VR simulation).

$$1.04 * K = 0.1 \tag{2}$$

$$0.34 * K = 0.1 \tag{3}$$

That can be generalized by the following control rule:

$$X * K = x \tag{4}$$

Where X is the output from the Gyroscope after filtering and integration (each variable is expressed in radians) and x is the final position on the on the VR simulator. Since it is in 2D, only two variables of X have been taken into account. They have been implemented on Simulink as shown in Fig. 13 for the Wrist and in Fig. 14 for the Torso.

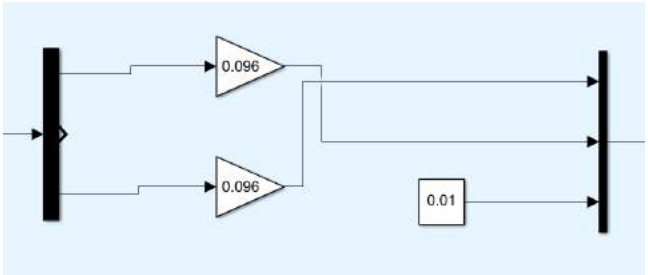

Figure 13: Wrist Range of Motion

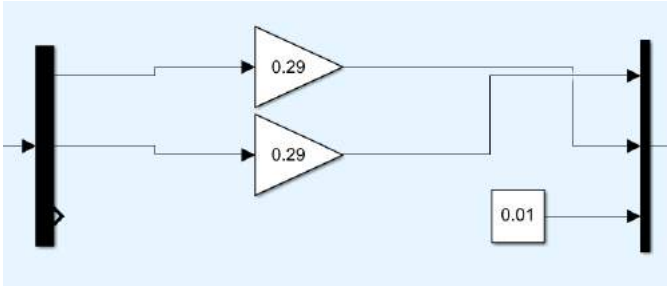

Figure 14: Torso Range of Motion

3.4.1 Wrist

In the Wrist's model only X and Z coordinates of the sensor's output are used in the pre-processing phase (Integration and FIR filter) because we decided to control the cursor through a rotation of the gyroscope around the *x-axis* and the *z-axis*. This allows the flexion/extension and ulnar/radial deviation of the wrist [Figure 15]. To perform these movements up to an amplitude of 60 degree we used a gain of **K= 0.096**.
After the definition of the Range of motion, the movement is mapped on the virtual world. The flexion/extension of the wrist moves the cursor on the *y-axis* of the VR simulator while the ulnar/radial deviation moves the cursor on the *x-axis*. The Z input component of the Simulator block is set to a fixed value: 0.01.

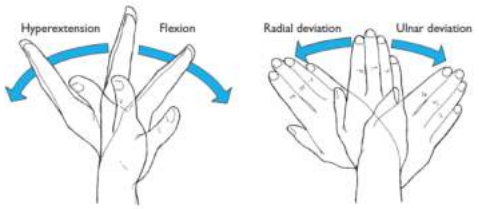

Figure 15: Wrist movements

3.4.2 Torso

In the Torso's model, only the X and Y coordinates of the sensor's output are used in the pre-processing phase. Using this part of the body, we decided to control the cursor through a rotation around the *x-axis* and the *y-axis* of the gyroscope. In such a case, the movements performed by the Torso are flexion/extension and rotation. [Figure 16].

To execute these movements up to an amplitude of 20 degrees we used a gain of **K= 0.29**.
After the definition of the Range of motion, the movement is mapped on the virtual world. The flexion/extension of the torso moves the cursor on the *y-axis* of the VR simulator while the rotation moves the cursor on the *x-axis*. Again, The Z input component of the Simulator block is set to a fixed value: 0.01.

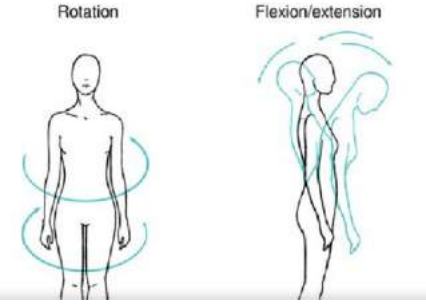

Figure 16: Torso movements

3.5 Mapping to a circle

In this model's subsystem, data are modified so that the cursor can only move inside the circle depicted on the simulation environment. This behavior is implemented through a simple Matlab function.

Input

- x_{in}, y_{in}, z_{in}

Output

- $X_{out}, Y_{out}, Z_{out}$

We first calculate the angle defined by X_{in} and Y_{in}: $\theta = atan2(Y_{in}, X_{in})$.
Then, if one of the coordinates falls outside of the small circle of balls, (that has radius=0.1), we impose it to be on the circumference. In any other case it we don't modify it.

In formulas:

$$\theta = atan2(Y_{in}, X_{in})$$

if $|X_{in}| > |0.1cos(\theta)$ **then**
$x_{out} = 0.1cos(\theta)$
end if
if $|Y_{in}| > |0.1sin(\theta)$ **then**
$y_{out} = 0.1sin(\theta)$
end if

The result is shown in Figure 17.

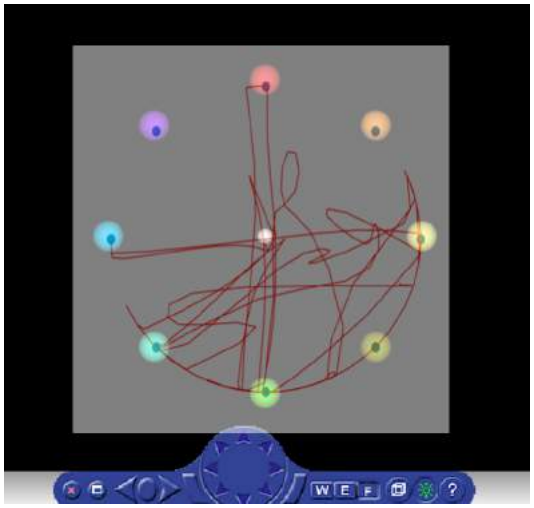

Figure 17: Mapping effect

4 System testing and results

4.1 System testing

Since our control scheme varies with reference to the different body parts the user wants to use, we created a simple GUI in order to test our project on both the wrist and the torso. The GUI was created using the Guide command [1] ans is shown in Figure 18. By clicking on one of the buttons, the relative Simulink model is opened and run.

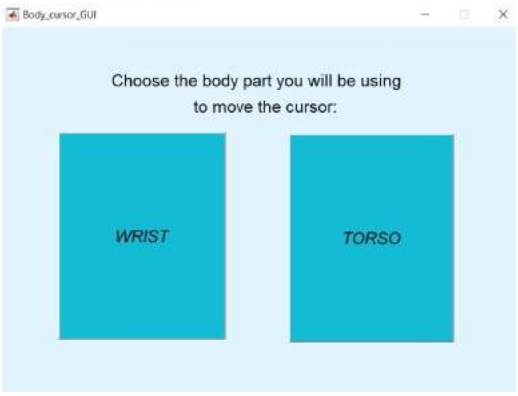

Figure 18: GUI

4.2 Results

The program has been tested with the smartphone attached on the wrist and on the torso, according to what we said above.
Hereby are shown the plots of

- the unfiltered data (Figures 22 and 26)
- the filtered data (Figures 23 and 27)
- the data before the circle constraint (Figures 24 and 28). The z value is already set at a constant value since we're working with a 2D simulation.
- the final data (Figures 25 and 29). It is evident from these last pictures that the signal is cut at 0.1.

The movement data was plotted by using a small code, called PlotData.m, which is a modification of the assignment 0 (that can be found on the same Github repo of this assignment).
Test's results executed on the wrist and on the torso are shown in Figures 19, 20, 21.

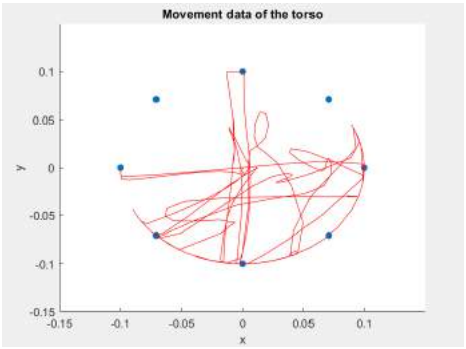

Figure 19: Torso trial: mapping

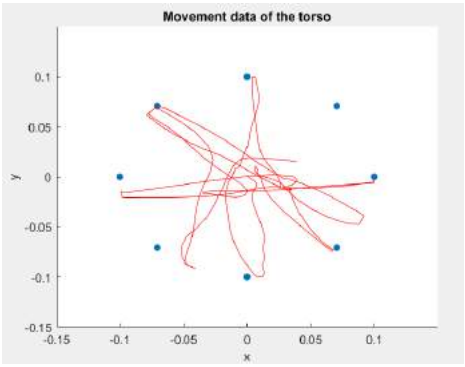

Figure 20: Torso trial

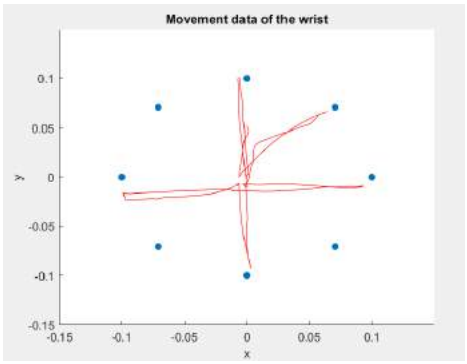

Figure 21: Wrist trial

Figure 22: Unfiltered torso data

Figure 23: Filtered torso data

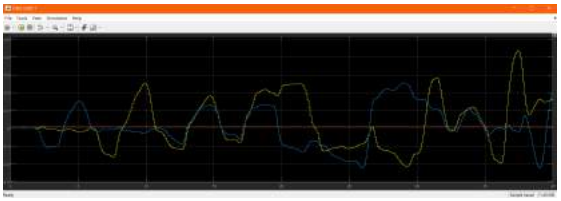

Figure 24: Torso data before circle constraint

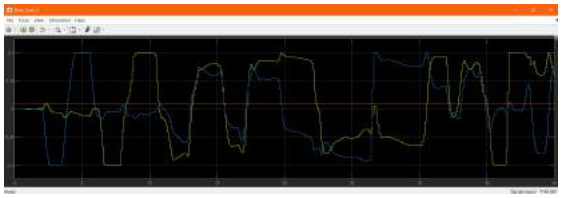

Figure 25: Final torso data

Figure 26: Unfiltered wrist data

Figure 27: Filtered wrist data

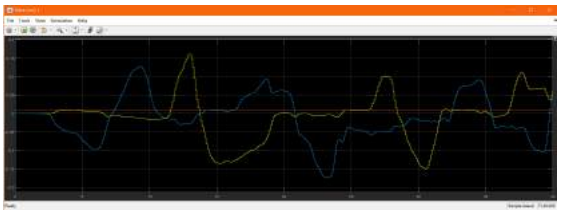

Figure 28: Wrist data before circle constraint

Figure 29: Final wrist data

4.3 Limitations and further improvements

Logically, our program has limitations and may be improved. As previously said, the gyroscope can detect changes in the rotation (Attitude) of the phone over the three axes and each rotation measurement is based on the previous estimate (it is a relative measurement approach). This means that there is an "initial drift" and, over time, errors will add up, as represented in Figure 30. Further shifting is also caused by quick changes in acceleration, such as small impacts on the device.

In order to avoid this problem, there are two possible approaches: either combin-

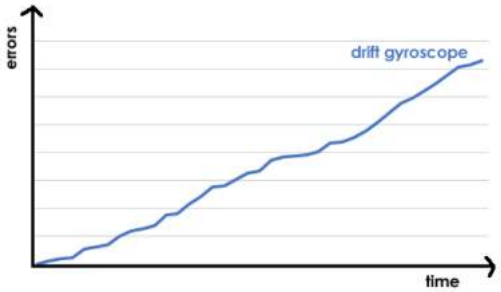

Figure 30: Drift accumulated over time

ing the gyroscope signal with the accelerometer (which includes all the initialization problem claimed before) or using an automatic reset during idle time.

We tried implementing an automatic reset during idle time. We did so by using the

Detect Change Simulink block to confront both desired input signals (x,t for the torso and x,z for the wrist). We compared data from subsequent time instants: if the phone was still for a certain number of sampling instants, the phone was considered static. When the idle detection occurred, a zero-reset on the integrator was triggered, and the cursor was set in the zero position. This worked mainly for the wrist, since it's more difficult to keep the smartphone steady on the torso. However, this approach worked well at small sampling frequencies (i.e. 10Hz) but was inapplicable at higher frequencies, so we removed it.

Additionally, even if calibration was actually implemented in the correct way, if there was no voluntary re-calibration at a certain point of the simulation, the errors would still increase in time. Moreover, during re-calibration it's important to stay still in a suitable position, to prevent the zero of the simulation to be set in an uncomfortable position for the wrist/torso. The patient should therefore be instructed to do so.

As another improvement, the GUI may be enhanced by adding other body positions (i.e. giving the possibility to place the smartphone on other points of the body) for further medical and diagnostic purposes. In order to do this, one would need to add a new file that has a modified gain value (to improve the sensibility of the motion according to the chosen body position) and also has the right vector components of the gyroscope to be used as input for VR Sink.

Finally, we could also improve the simulation in order to transform the project into a more engaging game aimed at rehabilitation (exergame).

References

[1] *Create a simple app using GUIDe*. MATLAB. 2019. URL: https://it.mathworks.com/help/matlab/creating_guis/about-the-simple-guide-gui-example.html.

[2] *Getting Started with Android Devices*. MATLAB. 2019. URL: https://it.mathworks.com/help/supportpkg/android/examples/getting-started-with-android-devices.html.

[3] *Install Support for Android Devices*. MATLAB. 2019. URL: https://it.mathworks.com/help/supportpkg/android/ug/install-support-for-android-devices.html.